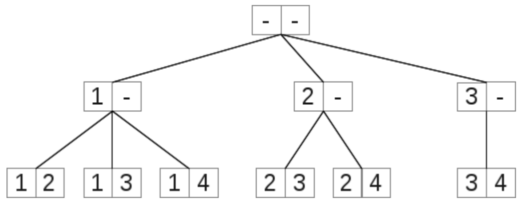
Introduction to Bioinformatics Algorithms
Homework 5

Saad Mneimneh, Computer Science, Hunter College of CUNY

Problem 1: Partial digest with "Branch and Bound"

(a) Given a set of size n, generate all subsets of size $k \leq n$. Think of the elements as being $\{1, 2, \ldots, n\}$. For example, the following tree illustrates all subsets when $k = 2$ and $n = 4$ (the leaves are at level k).



Observe that for each node in the tree, if the largest value is a and the level is l, then $n - a \geq k - l$ (otherwise the node cannot have any descendants). Using a function similar to the one covered in HW1, you can generate all subsets of a given size without having to explicitly generate the tree.

(b) Implement the brute force algorithm for the partial digest problem. You will need to generate all subsets of $L - \{\max L\}$ of size $n-2$ (check the pseudocode). You will also be able to prune certain subtrees if a partial subset cannot be extended, e.g. some distance is not in L. For this, adapt an implement of a *next* and a *skip* on the tree as explained in class for the branch-and-bound technique. Try the brute force algorithm with and without the pruning on a large example, e.g. generate a set of n random integers X and their pairwise distances L, then try to retrieve X from L.

Problem 2: Sorting by reversals

The algorithm on page 133 repeatedly chooses a reversal ρ that minimizes the number of breakpoints $b(\pi\rho)$ until the number of breakpoints is 0. The analysis following the presentation of this algorithm gives an approximation factor of 4.

(a) Show by example that when all strips are increasing, it is possible to choose a ρ as described above such that $\pi\rho$ has one decreasing strip, which when reversed, all strips are increasing again.

(b) Suggest a modification to the algorithm so that if every strip is increasing, there will be a decreasing strip in each of the next two steps. How does that affect the approximation factor?

Problem 3: Sequecing
Do problem 8.6 in the book.

Problem 4: Shortest superstring (optional)
The greedy algorithm for shortest covering string described in class essentially does this: repeatedly merge a pair of strings with maximum overlap until one string remains. Define the compression of the algorithm as the number of symbols saved compared to plainly concatenating all strings. Prove that the greedy algorithm achieves at least 1/2 the compression of the optimal superstring.